CLIENT TrustInSoft

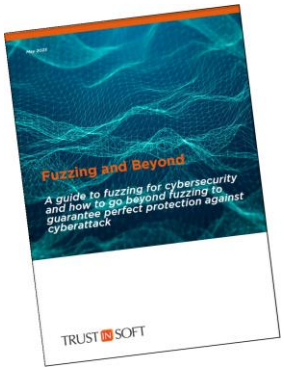
PROJECT White paper on the software practice of fuzzing

OBJECTIVE Create a white paper that explains fuzzing and how formal methods can be used to enhance its benefits, overcome its limitations, and ensure far greater cybersecurity in software products.

COPY EXCERPT

Fuzzing and Beyond

A guide to fuzzing for cybersecurity and how to go beyond fuzzing to guarantee perfect protection against cyberattack



Call or write CopyEngineer to receive a PDF of the complete white paper.

Or view/download it online at: bit.ly/TrustInSoft-WP-Fuzzing

The rapid growth in the connectivity of software-driven products to Wi-Fi and cellular networks and the internet—in industries that include consumer electronics, smart home and office, IoT, industrial automation, automotive, aerospace and defense—has been forecast to continue to accelerate for the foreseeable future. [i, ii]

That growth has engendered a corresponding expansion in attack surfaces and, subsequently, an increasing interest in exploiting those attack surfaces on the part of hackers and governments. In short, the cyber threat is here to stay.

Countering that threat has become critically important to businesses and has resulted in a growing interest in the use of *fuzzing* on the part of both software development organizations and security specialists.

Fuzzing is a software testing technique that applies vast numbers of input combinations to a target program very rapidly, in an automated manner. By generating these inputs semi-randomly, fuzzing can test combinations the developer may not have anticipated while saving the developer the tedium of manually defining individual test cases. The goal is to reveal hard-to-find vulnerabilities that are rarely caught by conventional software testing.

Hackers also frequently use fuzzing tools to find “loopholes” in code—untested input values that create unexpected behavior they can exploit remotely. They wait while their fuzzer applies millions of random inputs until, finally, it uncovers a flaw that suits their purposes.

To unleash havoc, a hacker need only find a single vulnerability the developer failed to correct before release. That puts software vendors at a distinct disadvantage.

While testing for corner cases is a best practice, development teams need to go further. Under time-to-market pressure, they need new methodologies for efficient verification. Unlike hackers, they can't be satisfied with stumbling upon one exploitable vulnerability. They need to find and fix all the vulnerabilities they can while maintaining their release schedule.

A large proportion of issues in C/C++ code that are exploited by hackers are so-called undefined behavior. Undefined behavior is a technical term that includes all sorts of runtime errors such as buffer overflows, division by zero, null pointers, etc. Fuzzing is a great first step for uncovering undefined behavior that normal testing is not designed to catch. However, even the best fuzzers are not designed to catch every vulnerability…

---

[i] Hyper Connectivity Market Forecast 2022-2030, Precedence Research, September 2022.

[ii] Internet of Things Connectivity Market, Emergen Research, June 2022.